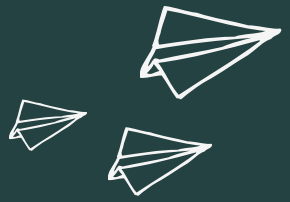


go 语 言 编 程

目录页

第四章
Part four

1 数组

2 切片

3 映射

4.1 数组

数组是一系列相同类型数据的集合，数组中包含的每个数据被称为数组元素。一个数组包含的元素个数称为数组的长度，数组长度是固定的。一个数组可以由零个或多个元素组成。

第四章 数组、切片和映射

1 数组声明

数组的声明的一般形式： var 数组名称 [长度] 类型

2 简短声明

与变量的简短声明一样，数组也可以简短声明。

3 元素访问

数组的每个元素通过索引下标来访问。

4 数组遍历

两种遍历方式，其中range表达式遍历有两个返回值，第一个是索引，第二个是元素的值。

第四章 数组、切片和映射

5 作为函数参数

在Go语言中，数组作为函数的参数仍然是值传递，虽然可以使用数组的指针来代替，但是改变不了数组长度。这时，我们通常使用切片slice来替代数组。

6 数组比较

如果数组元素的类型是可比较的，那么这个数组也是可的比较。

4.2 数组切片

数组的长度在定义之后无法修改，数组是值类型。显然这无法满足开发者的某些需求。Go语言提供了数组切片（slice）来弥补数组的不足。

1 声明数组切片

数组切片的声明与数组声明类似，唯一的区别是无需指定长度。

var 切片名称 [ ] 类型

第四章 数组、切片和映射

2 基于数组以及基于切片创建切片

3 直接创建切片

Go语言提供了内置的make()函数可以灵活的创建切片。

make([]类型，长度)

make([]类型，长度，容量)

如果省略容量，那么容量将等于长度 。 内置的len()函数可以获取长度，内置的cap()函数可以获取容量。

与数组相同，slice操作不能超出len指定的范围。

4 切片的遍历，与数组的遍历相同。

第四章 数组、切片和映射

5 切片不能比较

6 判断切片是否为空，使用内置的len(s)==0来判断slice是否为空。

7 追加元素 ，内置的append()函数用于向slice追加元素。可以直接追加元素，也可以追加一个slice。注意slice后有...，否则有语法错误。

8 切片复制，内置的copy()函数用于数组切片的复制。复制时，无需考虑目标数组和源数组的长度。

9 作为函数参数时切片与数组的区别

4.3 映射

映射是一个无序的键值对集合。映射中所有的键都有相同的类型，所有的值也有着相同的类型，但是键和值之间可以是不同的数据类型。

1 声明映射的一般语法格式：

var 映射名称 map[key] value

只声明一个映射，初始值是nil。不能向一个nil值的映射存入元素

2 创建映射,内置的make()函数可以创建映射，创建后可以给元素赋值。

3 元素的赋值和删除

元素可以直接赋值，内置的delete()函数可以删除元素，删除一个不存在的元素，不会造成错误。

4 查找元素

有时候需要直到对应的元素实在在映射中,如果元素类型是布尔类型，可能需要区分一个零值的元素和因为元素不存在而返回的零值。可以这样使用：

```
if v,ok:=map[key]; !ok{  /* 不存在时的处理*/ }
```

第四章 数组、切片和映射

5 遍历映射

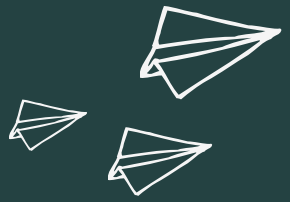
遍历映射使用range来实现，但是由于map是无序的集合，所以每次遍历的顺序可能不一样。

6 映射不可比较

和slice一样，映射不可比较。唯一例外的是可以和nil比较。

7 不能对映射元素取址操作

例如： &map[0];// 是错误的，map不能取址操作



THANKS

感谢聆听，期待反馈